INSTITUTO DE GESTÃO EM TECNOLOGIA DA INFORMAÇÃO

JAVA AUTHENTICATION E AUTHORIZATION SERVICE COMO MECANISMO DE SEGURANÇA E CONTROLE DE ACESSO EM APLICAÇÕES WEB

Luciano Henrique Teixeira Bianchi
Jacqueline Fonseca
Projeto de Aplicações Java

Resumo

Em um cenário onde as informações precisam ser acessadas a qualquer momento e de qualquer lugar as aplicações web tornam-se cada vez mais utilizados por proporcionar tal agilidade e facilidade. Nesse contexto é indispensável que as informações possam ser acessadas de forma segura e confiável, sendo que os conceitos de autenticação e autorização ganham papel importante no controle de acesso. O JAAS (Java Authentication and Authorization Service) é a API padrão do Java para controle de acesso e autorização em aplicações JavaEE. (JAVEAUX, 2009). Mecanismos de segurança como este é um recurso valioso nesse contexto baseado em identidade e role(s) garantindo níveis de segurança, baixo acoplamento e portabilidade. O objetivo é mostrar como é possível desenvolver mecanismos de autenticação e autorização mais robustos, flexíveis e consistentes utilizando o JAAS.

Palavras-chave: Autenticação. Autorização. Java Authentication. Authorization Service. JAAS. Aplicações Web.

Abstract

In a scenario where information needs to be accessed at any time and from anywhere web applications are increasingly used to provide such agility and ease. In this context it is indispensable that the information can be accessed in a secure and reliable way, and the concepts of authentication and authorization gain an important role in access control. Java Authentication and Authorization Service (JAAS) is the default Java API for access control and authorization in JavaEE applications. (JAVEAUX, 2009). Security mechanisms like this are a valuable resource in this context based on identity and role (s) ensuring levels of security, low coupling and portability. The goal is to show how it is possible to develop more robust, flexible and consistent authentication and authorization mechanisms using JAAS.

Keywords*: Authentication. Authorization.* Java Authentication. Authorization Service. JAAS. Web Applications.

1. INTRODUÇÃO

A segurança da informação é um tema de grande relevância no desenvolvimento de sistemas web onde é muito comum encontrarmos aplicações que restringem o acesso de usuários ou processos a apenas alguns de seus recursos, no entanto implementar esse tipo de controle não é uma tarefa fácil. A existência de diversos mecanismos de segurança para que se possa identificar as credenciais de um usuário no sistema como por exemplo através de Banco de Dados (Oracle, MySQL, SQL Server), Domain Servers, Lightweight Access Directory Protocol (LDAP), Biometria, dentre outros aumenta significativamente a necessidade de alterar o código a nível de aplicação para suportar cada um desses mecanismos.

O Java Authentication and Authorization Service (JAAS) é um framework que se propõe a resolver exatamente isso colocando uma camada de abstração entre o aplicativo e os diferentes mecanismos de segurança.

Diante do exposto, o objetivo é explorar os recursos do JAAS para que se possa aplicar os conceitos de autenticação e autorização em aplicações web de forma mais coesa e coerente, afim de promover maior segurança e independência através do conceito de plugabilidade proposto pelo framework alinhado as melhores práticas de desenvolvimento.

A organização desse artigo apresenta-se da seguinte forma: A seção dois apresentará a metodologia utilizada para o desenvolvimento da pesquisa. A seção três apresentará alguns conceitos de Segurança da Informação, Autenticação e Autorização para que se possa detalhar os mecanismos e possibilidades presentes no JAAS.

A seção quatro apresentará o framework JAAS, seu funcionamento, características importantes, e análise dos tipos de autenticação e autorização presentes no framework.

A seção cinco apresentará um estudo de caso de uma aplicação real do Superior Tribunal de Justiça que utiliza um dos mecanismos presentes no JAAS para realizar o controle de acesso ao sistema.

Por fim, a última seção apresentará uma análise dos principais pontos positivos e negativos ao se utilizar o framework e as considerações finais.

2. METODOLOGIA

Para a elaboração desse artigo foi feito um estudo bibliográfico extensivo sobre as melhores práticas em segurança da informação aliada ao estudo da documentação oficial do JAAS disponibilizada pela Oracle e de outras fontes de pesquisa não oficiais com o intuito de explorar ao máximo todo o conteúdo relevante já pesquisado sobre o tema.

Adicionalmente foi feito um estudo de caso na aplicação Portal da Intimação Eletrônica do Superior Tribunal de Justiça (STJ) entre os dias 30/01/2017 e 02/02/2017. Durante esse período foi feito junto da equipe de desenvolvimento de software do STJ a engenharia reversa referente a camada de segurança da aplicação com o intuito de delimitar um pouco mais o universo do JAAS explorando uma de suas unidades implementada em uma aplicação real. “Engenharia Reversa (RE) é um processo de medição, análise e teste para reconstruir a imagem espelhada de um objeto ou recuperar um evento passado” (WANG, 2010, p. 1).

Na posição de Lüdke e Andre (1986), o estudo de caso como estratégia de pesquisa é o estudo de um caso, simples e específico ou complexo e abstrato e deve ser sempre bem delimitado. Pode ser semelhante a outros, mas é também distinto, pois tem um interesse próprio, único, particular e representa um potencial na educação. Destacam em seus estudos as características de casos naturalísticos, ricos em dados descritivos, com um plano aberto e flexível que focaliza a realidade de modo complexo e contextualizado.

Tendo em conta a posição dos autores apresentados, o estudo de caso pretende investigar, como uma unidade, as características importantes como objeto de estudo para a pesquisa.

3. REVISÃO DE LITERATURA

Quando se fala em Segurança da Informação pode-se utilizar a representação da "pirâmide do conhecimento" para elucidar que um dado isolado não representa valor, porém um dado processado gera informação que gera conhecimento.

Figura 1: Pirâmide do conhecimento.
Fonte: JUNIOR, 2012.

"A informação é elemento essencial para todos os processos de negócio da organização, sendo, portanto, um bem ou ativo de grande valor" (CAMPOS, 2006, p. 9). Pode-se constatar que a informação é um ativo essencial para os negócios da organização, sendo necessário proteger essa informação de maneira adequada.

3.1. SEGURANÇA DA INFORMAÇÃO

Atua na garantia da confidencialidade, integridade e disponibilidade dos dados não importando a forma que eles se apresentem, seja em formato eletrônico, papel, ou outros formatos.

Em aplicações distribuídas através de redes abertas os usuários interconectados com essa aplicação podem estar compartilhando da mesma rede com outros usuários que não estejam autorizados a acessar determinados sistemas. Surge assim a necessidade de identificar e autenticar os usuários capazes de acessar o sistema bem como

verificar quais são os seus privilégios de acesso a determinados recursos, esse processo é feito em duas etapas: Autenticação e Autorização.

3.2. AUTENTICAÇÃO

Essa é a etapa em que o usuário deverá provar a sua identidade, o que pode ser feito através de três possíveis formas:

- Conhecimento – o sistema requer uma informação da qual o usuário possua conhecimento, como por exemplo o fornecimento de uma senha.
- Propriedade – o sistema exige a apresentação de algo concreto, como por exemplo um cartão eletrônico ou um certificado digital.
- Característica – o sistema busca através de algum dispositivo ler determinada característica física do usuário, como por exemplo a sua digital ou retina, comparando assim com uma imagem previamente cadastrada.

3.3. AUTORIZAÇÃO

Autorização é a capacidade de validar após a autenticação, em uma lista de acesso pré-definida, se algo, ou alguém, possui permissões para realizar ações com dados em um sistema da informação. A autorização sempre ocorre após a autenticação. (RESS, 2011).

Esse mecanismo de segurança usualmente é implementado através de três possíveis formas:

- Access Control List (ACL) – cada objeto tem uma lista de (ação, lista de usuários).
- Capabilities – contraparte do sistema ACL, um usuário é associado a uma lista de (ação, lista de objetos)

- Access Control Matrix – matriz onde as listas são compostas por usuários, as colunas por objetos e os elementos são lista de permissões. Pode ser representado tanto na ACL como na Capabilities.

4. APRESENTAÇÃO DA PESQUISA

4.1. PLUGGABLE AUTHENTICATION MODULE

O JAAS fornece uma estrutura de autenticação plugável padrão (PAM), que é um mecanismo que integra vários esquemas de autenticação de baixo nível em uma API de alto nível que permite que as aplicações que dependam da autenticação sejam escritas independentemente do esquema de autenticação subjacente.

Com isso o desenvolvedor pode alavancar uma série de módulos existentes diminuindo-se assim os esforços de autenticação de baixo nível em um todo mais gerenciável contribuindo para simplificar o mecanismo de segurança.

É importante ressaltar que o JAAS é conceitualmente análogo a PAM, porém não existe um módulo PAM disponível no framework, dessa forma caso se queira utilizar os módulos PAM com o JAAS é necessária uma bridge, que seriam interfaces responsáveis por realizar essa ligação. Há alguns projetos disponíveis como o JAAS-PAM e o JPAM.

4.2. API JAAS

O JAAS foi integrado de forma definitiva ao SDK (Java Standard Edition Development Kit) a partir da versão 1.4.

As classes e interfaces chave do JAAS são tipicamente divididas em três grupos:

Quadro 1: Classes e Interfaces chave.

Comum	Subject, Principal

Autenticação	LoginContext, LoginModule, CallbackHandler
Autorização	Policy, PrivilegedAction

Fonte: Elaborado pelo autor.

4.2.1. SUBJECT

Representa a entidade autenticada, pode ser um usuário final, web service, dispositivo, ou qualquer outro processo.

4.2.2. PRINCIPALS

Representa a identidade do Subject, um Subject pode possuir um conjunto de Principals que o identifique, onde cada Principal pode representar uma característica específica como por exemplo Identidade, Seguro Social, CPF, que reunidas representam um sujeito do mundo real.

4.2.3. LOGINCONTEXT

Representa onde a funcionalidade dinâmica do JAAS entra em ação, de acordo com Bharghav e Kumar (2010) é a classe mais importante do JAAS que fornece os métodos básicos usados para autenticar assuntos e pavimenta uma maneira de desenvolver um aplicativo que é independente da tecnologia de autenticação subjacente. O seu construtor recebe o nome da configuração a ser carregada, definida normalmente em um arquivo de texto, que por sua vez informa ao LoginContext qual LoginModule utilizar durante o processo de login.

4.2.4. LOGINMODULE

Interface responsável por definir o comportamento a ser implementado pelas classes que irão realizar a autenticação. O JAAS já possui um conjunto de LoginModules prontos para uso, como por exemplo JndiLoginModule, Krb5LoginModule, NTLoginModule, sendo que todos esses módulos possuem um conjunto de implementações de Principal correspondente. É possível criar um LoginModule próprio, para isso basta implementar os métodos definidos na interface LoginModule.

4.2.5. CALLBACKHANDLER

Interface responsável por transportar as informações de autenticação necessárias permitindo que um LoginModule as receba enquanto permanece independente do mecanismo de interação atual. Também é possível criar um CallbackHandler próprio, para isso basta implementar o único método definido na interface CallbackHandler.

4.2.6. POLICY

Representa a diretiva de controle de acesso ao sistema. Como padrão fornece uma implementação de subclasse baseada em arquivos, suportando também consultas baseadas em Principal.

4.2.7. PRIVILEGEDACTION

Toda operação que necessite ser executada com verificações de autorização baseadas em Principals deve implementar o método run dessa interface.

4.3. FUNCIONAMENTO DO JAAS

Totalmente compatível com o padrão JEE e portável em qualquer servidor JEE a estrutura do JAAS é basicamente composta de uma Identidade, que se refere ao mecanismo de Autenticação, e de Role(s) que são perfis de acesso que se referem ao mecanismo de Autorização.
Através do mecanismo de Autorização do JAAS é possível restringir de forma declarativa ou programática o acesso a URLs, diretórios, métodos de componente e conteúdo, o que possibilita o desacoplamento entre segurança e regras de negócio.

Através do uso de uma estrutura extensível de Interfaces Provedoras de Serviços (SPIs) plugáveis, o JAAS é independente de implementação. SPIs são um conjunto de classes abstratas e interfaces para as quais implementações específicas são desenvolvidas.

4.3.1. PLUGABILIDADE E EMPILHAMENTO

A camada de aplicação lida principalmente com um LoginContext que se comunica com um conjunto de um ou mais LoginModules configurados dinamicamente que identificam a autenticação atual usando a infraestrutura de segurança apropriada.

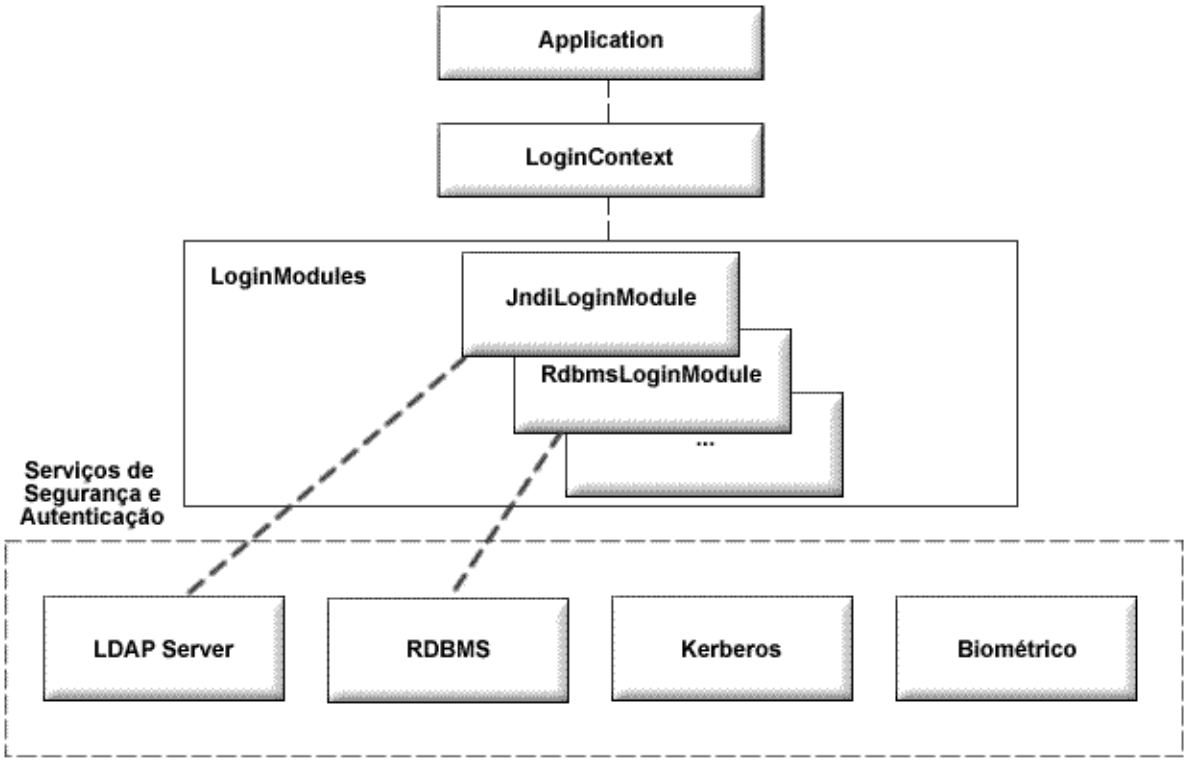

Figura 2: Arquitetura de alto nível JAAS.
Fonte: Elaborado pelo autor.

Além de ser plugável o JAAS é empilhável no contexto de um único login. Um conjunto de módulos de segurança pode ser empilhado um sobre o outro, cada um invocado em ordem e cada um interagindo com uma infraestrutura de segurança diferente. Com isso é possível criar estruturas complexas e independentes como por exemplo em um mesmo processo de login realizar autenticação em um servidor LDAP e seguidamente realizar uma autenticação biométrica, concedendo o acesso apenas com êxito nos dois mecanismos.

4.3.2. WORKFLOW DE AUTORIZAÇÃO

O JAAS possui um workflow simples e eficiente para que uma entidade autenticada (Subject) possa utilizar os recursos protegidos.

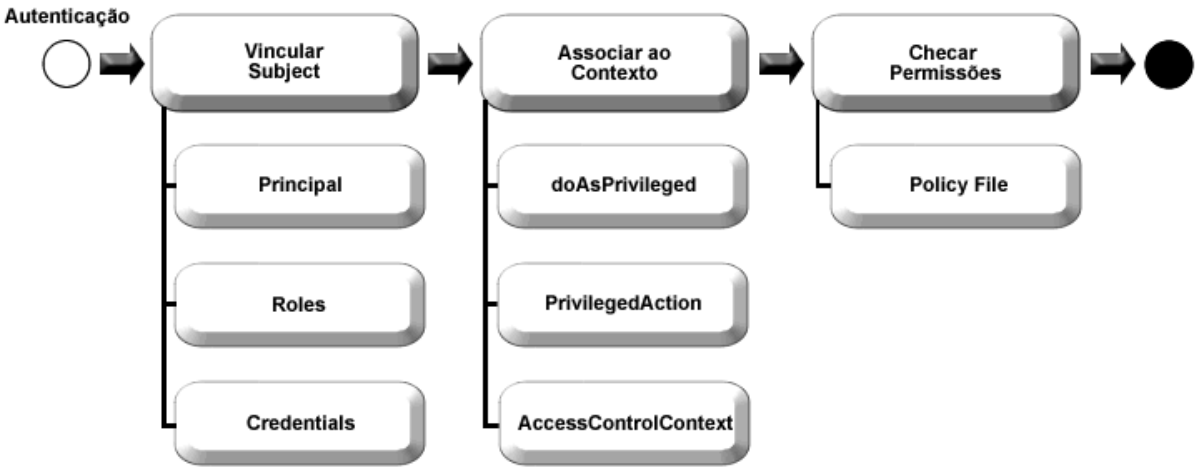

Figura 3: Workflow de autorização.
Fonte: Elaborado pelo autor.

Após a etapa de autenticação é criado um Subject, que representa a entidade autenticada, a esse Subject pode ser vinculado:

- Um conjunto de Principals, que representam a identidade de Subject, e devem implementar as interfaces java.security.Principal e java.io.Serializable. (ORACLE, 2017).
- Um conjunto de Roles, que representam os perfis de acesso do Subject, uma Role é estruturalmente um Principal, devendo implementar as mesmas interfaces.
- Um conjunto de Credentials, as classes de credenciais públicas e privadas não fazem parte da biblioteca de classes principais do JAAS. Qualquer classe pode representar uma Credential. Os desenvolvedores, no entanto, podem optar por ter suas classes de credenciais implementadas por duas interfaces: Refreshable e Destroyable. (ORACLE, 2017).

Para que se possa proteger uma determinada operação, é necessário que seja implementada a interface PrivilegedAction. O Subject para poder ter acesso a essa operação protegida deve invocar o método doAsPrivileged informando qual contexto de controle de acesso ele deve ser associado e qual PrivilegedAction deseja executar. O objeto AccessControlContext informado no método doAsPrivileged que é o responsável

por checar no arquivo de políticas de acesso se o Principal do Subject autenticado possui a permissão necessária para a execução dessa operação protegida.
Todo esse processo alcança o efeito de ter a ação executada pelo sujeito.

4.3.3. DOASPRIVILEGED X DOAS

Os métodos doAsPrivileged se comportam exatamente iguais aos métodos doAs, exceto que em vez de associar o Subject fornecido com AccessControlContext do Thread atual, eles usam o AccessControlContext fornecido. Desta forma, as ações podem ser restringidas pelo AccessControlContext diferente do atual. (ORACLE, 2017).

4.3.4. SEGURANÇA DECLARATIVA X PROGRAMÁTICA

O JAAS suporta ambos os mecanismos, sendo possível trabalhar em conjunto ou separadamente. A segurança declarativa é feita com o uso de anotações, sendo as principais:

- @RolesAllowed – define um range de roles que possuem acesso
- @PermitAll – concede acesso a todos
- @DenyAll – nega acesso a todos

Já para a segurança programática, é necessário obter o contexto atual da aplicação e realizar a verificação com a invocação de métodos específicos para checar se o usuário está associado a determinada role.

5. ESTUDO DE CASO

5.1. APRESENTAÇÃO

Como estudo de caso para verificação prática do framework JAAS utilizado como mecanismo de segurança em uma aplicação implantada, foi realizado o acompanhamento técnico do sistema Portal da Intimação Eletrônica do Superior Tribunal de Justiça (STJ) entre os dias 30/01/2017 e 02/02/2017 afim de analisar quais técnicas do framework JAAS foram utilizadas na camada de segurança da aplicação assim como auferir os possíveis benefícios, desvantagens e limitações encontradas com o uso do JAAS.

5.2. DETALHAMENTO DE REQUISITOS

Para o desenvolvimento da aplicação Intimação Eletrônica foram utilizados os frameworks EJB 3.1, JSF 2.1 e JEE 6, e o servidor de aplicação JBoss EAP 6.2.

5.3. DETALHAMENTO TÉCNICO

A etapa de Autenticação foi desenvolvida através da criação de um LoginModule próprio, esse LoginModule é responsável por verificar no banco de dados se as credenciais fornecidas no formulário de login são válidas.
O JAAS disponibiliza 5 possibilidades para a aplicação apresentar a autenticação aos usuários:

- None – não é realizado nenhum tipo de autenticação;
- Basic – o browser apresenta uma janela ao usuário para entrada das credenciais;
- Digest – semelhante ao Basic, porém as credenciais são criptografadas de acordo com o algoritmo escolhido no arquivo de configuração;
- Certificate – baseada em certificados públicos e privados;
- Form – baseada em um formulário web.

A aplicação utilizou a opção do Form, sendo que essa é a forma mais comum vista em aplicações web.

A configuração do Form deve ser realizada no arquivo de configuração do Web Container, nesse caso o arquivo web.xml conforme visto abaixo:

```
<!-- Login page -->
 <login-config>
    <auth-method>FORM</auth-method>
    <realm-name>intimacaoJaasDB2Realm</realm-name>
    <form-login-config>
        <form-login-page>/public/login.xhtml</form-login-page>
        <form-error-page>/public/loginError.xhtml</form-error-page>
    </form-login-config>
</login-config>
```

Figura 4: Diretivas de autenticação.
Fonte: *print screen* do código fonte da aplicação Portal da Intimação Eletrônica.

Conforme visto, nessa etapa é definido um nome (realm-name) que identifica qual será o domínio de segurança utilizado.
Essa configuração fica protegida dentro do Web Container, sendo necessário criar apenas uma referência na aplicação para o Web Container utilizado, nesse caso o arquivo jboss-web.xml é utilizado para referenciar o domínio de segurança.

```
<security-domains>
    <security-domain name="intimacaoJaasDB2Realm" cache-type="default">
        <authentication>
            <login-module code="br.jus.stj.jurisdicionado.intimacao.jaas.IntimacaoLoginModule" flag="required"/>
        </authentication>
    </security-domain>
```

Figura 5: Configuração do domínio de segurança.
Fonte: *print screen* do código fonte da aplicação Portal da Intimação Eletrônica.

No domínio de segurança é definido qual LoginModule será utilizado, nesse caso foi referenciado o próprio LoginModule criado.

O LoginModule deve implementar a interface javax.security.auth.spi.LoginModule e os seus 5 métodos:

- initialize – responsável por atribuir principalmente o Subject e o CallbackHandler, assim como inicializar qualquer outra classe auxiliar na etapa de autenticação;
- login – responsável por recuperar as credenciais através do CallbackHandler e realizar o processo de verificação, no caso da aplicação é feita uma consulta ao banco de dados comparando as credenciais informadas, esse método retorna um booleano em que caso verdadeiro a autenticação foi realizada com sucesso;
- commit – responsável por associar o conjunto de Principals e Roles ao Subject;
- abort – utilizado para abortar o processo de autenticação, esse método não é utilizado pela aplicação;
- logout – responsável por remover do Subject os Principals e Roles associados.

A etapa de Autorização é feita exclusivamente em cima das roles associadas ao Subject, para isso foi necessário criar todas as roles disponíveis no web.xml:

```
 <security-role>
    <role-name>ADMININT</role-name>
</security-role>
<security-role>
    <role-name>USERINT-V</role-name>
</security-role>
<security-role>
    <role-name>USERINT-C</role-name>
</security-role>
```

Figura 6: Definição das roles.
Fonte: *print screen* do código fonte da aplicação Portal da Intimação Eletrônica.

A aplicação realizou restrições de acesso a diretórios que podem ser criadas no próprio web.xml:

```
<!-- Protected Areas -->
 <security-constraint>
    <web-resource-collection>
        <web-resource-name>Somente admins</web-resource-name>
        <url-pattern>/protected/admin/*</url-pattern>
        <http-method>DELETE</http-method>
        <http-method>PUT</http-method>
        <http-method>HEAD</http-method>
        <http-method>OPTIONS</http-method>
        <http-method>GET</http-method>
        <http-method>POST</http-method>
        <http-method>TRACE</http-method>
    </web-resource-collection>
    <auth-constraint>
        <role-name>ADMININT</role-name>
    </auth-constraint>
</security-constraint>
<security-constraint>
    <web-resource-collection>
        <web-resource-name>Usuarios e Admins</web-resource-name>
        <url-pattern>/protected/user/*</url-pattern>
        <http-method>DELETE</http-method>
        <http-method>PUT</http-method>
        <http-method>HEAD</http-method>
        <http-method>OPTIONS</http-method>
        <http-method>GET</http-method>
        <http-method>POST</http-method>
        <http-method>TRACE</http-method>
    </web-resource-collection>
    <auth-constraint>
        <role-name>ADMININT</role-name>
        <role-name>USERINT-V</role-name>
        <role-name>USERINT-C</role-name>
    </auth-constraint>
</security-constraint>
```

Figura 7: Proteção de diretórios através de roles
Fonte: *print screen* do código fonte da aplicação Portal da Intimação Eletrônica.

Conforme visto, caso um usuário que não possua a role ADMININT por exemplo tente acessar alguma página contida no diretório /protected/admin/ o próprio web container vai ativar o mecanismo para restringir o acesso.

A aplicação também faz uso da verificação de segurança programaticamente, onde em pontos específicos uma controladora de sessão é acionada para saber se o usuário autenticado possui determinada role de acesso através do método getFacesContext().getExternalContext().isUserInRole(role).

Isso é possível devido ao contexto da aplicação ser JSF e a chamada ao LoginModule ser feita por um servlet, na controladora de login não é utilizado o LoginContext para

esse processo e sim o método login de HttpServletRequest, passando como parâmetro o login e senha informados no formulário.

Como o web container já sabe qual é o LoginModule que deve ser acionado, ao invocar o método login() o próprio container vai inicializar o processo de autenticação a partir do método initialize().

5.4. ANÁLISE TÉCNICA

A aplicação fez uso do framework JAAS de forma simples, porém eficiente, protegendo acesso a recursos diretamente no Web Container, delimitando o escopo da camada de segurança em poucas classes e integrando recursos do próprio web container com o JAAS na parte de autorização.

Caso houvesse necessidade de se passar além das credenciais básicas como login e senha alguma outra informação importante para o processo de autenticação, poderia ser criado um CallbackHandler próprio e utilizado o LoginContext para realizar a chamada do LoginModule passando o CallbackHandler criado.

6. CONSIDERAÇÕES FINAIS

Esse trabalho descreveu a utilização do JAAS como mecanismo para autenticar e autorizar usuários em uma aplicação web. Foi verificado que o JAAS fornece um modelo dinâmico, independente e extensível, oferecendo módulos previamente construídos assim como a flexibilidade de se criar os seus próprios mecanismos de login.

Os módulos de autenticação independentes da aplicação trazem grandes benefícios uma vez que usuários diferentes da aplicação podem se autenticar de forma distinta de acordo com suas necessidades. É possível alterar a maneira como o usuário autentica facilmente, apenas configurando o web container, sem que pare isso tenha que ser alterado código a nível de aplicação.

Como o JAAS é uma forma padrão de controle de acesso do Java, tecnologias futuras sérias irão suportar a API que permanecerá atualizada.

O JAAS por trabalhar análogo ao PAM mas não possuir um módulo PAM embutido pode frustrar as expectativas de quem deseja dentro de uma aplicação web escrita em Java e em execução em um ambiente Linux por exemplo autenticar um usuário como usuário de sistema.

Trabalhos futuros podem ser desenvolvidos no sentido de embutir um módulo PAM para permitir esse tipo de autenticação agregando ainda mais valor ao framework.

REFERÊNCIAS BIBLIOGRÁFICAS

WANG, Wego. **Reverse Engineering:** Technology of Reinventation. New York: CRC Press, 2010.

LÜDKE, M.; ANDRE, M. E. D. A. **Pesquisa em educação:** abordagens qualitativas. São Paulo: EPU, 1986.

CAMPOS, André L. N., **Sistema de Segurança da Informação:** Controlando os Riscos. Florianópolis: Visual Books, 2006.

BHARGAV, Abhay; KUMAR, B. V. **Secure Java:** For Web Application Development. New York: CRC Press, 2010.

RESS, Weber. **Começando em segurança**. 2011. Disponível em: https://msdn.microsoft.com/pt-br/library/ff716605.aspx#autorizacao. Acessado em: 30 jan. 2017.

ORACLE. **Java Authentication and Authorization Service (JAAS) Reference Guide**. 2017. Disponível em: http://docs.oracle.com/javase/8/docs/technotes/guides/security/jaas/JAASRefGuide.html. Acessado em: 01 fev. 2017.

JUNIOR, Helvio. **Clareza e produtividade na gestão do Firewall Aker 6.1.** 2012**.** Disponível em: http://www.helviojunior.com.br/it/security/clareza-e-produtividade-na-gestao-do-firewall-aker-6-1/. Acessado em: 30 jan. 2017.